

Creación e implementación de un protocolo de la Capa de Aplicación

Alma Rosa Páes Alcalá
Alejandro Valderrama Silva

Diciembre 9, 2019

Objetivo:

El objetivo de esta aplicación es llevar entretenimiento a gente que se quiere distraer de su cotidianidad a través de uno de los mejores temas de la historia: Pokemon. Pero no se queda ahí, nuestra aplicación también busca que los usuarios interesados puedan aprender un poco acerca de cómo diseñar y crear un protocolo TCP y maneras generales de ocupar bibliotecas de Python para ello. Esto es fácilmente lograble gracias a la documentación que proporcionamos y a nuestro código libre.

Una de las ventajas de esta aplicación es que es fácilmente adaptable a nuevas necesidades que surjan en el mercado ya que la estructura y la forma de comunicación no quedará obsoleta y puede aplicarse a distintas áreas sin problemas. Además esta estructura (base) puede ser utilizada en el desarrollo de nuevas aplicaciones que requieran el diseño de protocolos en TCP.

Aunado a esto, esta aplicación recopila la esencia de Pokemon; la cual consideramos que es capturar Pokemones sin importar sus características y así lograr conseguir el mejor Pokedex y convertirte en el mejor entrenador Pokemon. Que mejor que hacer esto con los Pokemones de la primera generación, los que inspiraron y generaron su éxito.

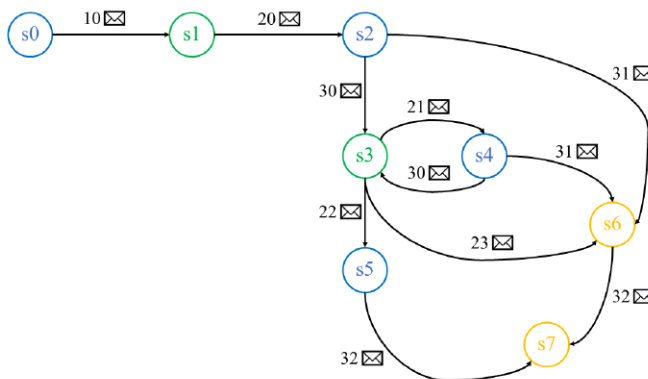
Repositorio:

Todo lo necesario para realizar lo mencionado en este documento, incluyendo instalar y jugar con la aplicación, se encuentran en la siguiente liga.

<https://github.com/AlmaPaes/Proyecto2-Pokemon>

Diseño del protocolo:

Como base del diseño del protocolo se tomo el autómata ya proporcionado, es decir:



Donde cada uno de los esta A los cuales los mensajes que le corresponden son los siguientes:

En común	
Código	Descripción
30	Sí.
31	No.
32	Terminando sesión.

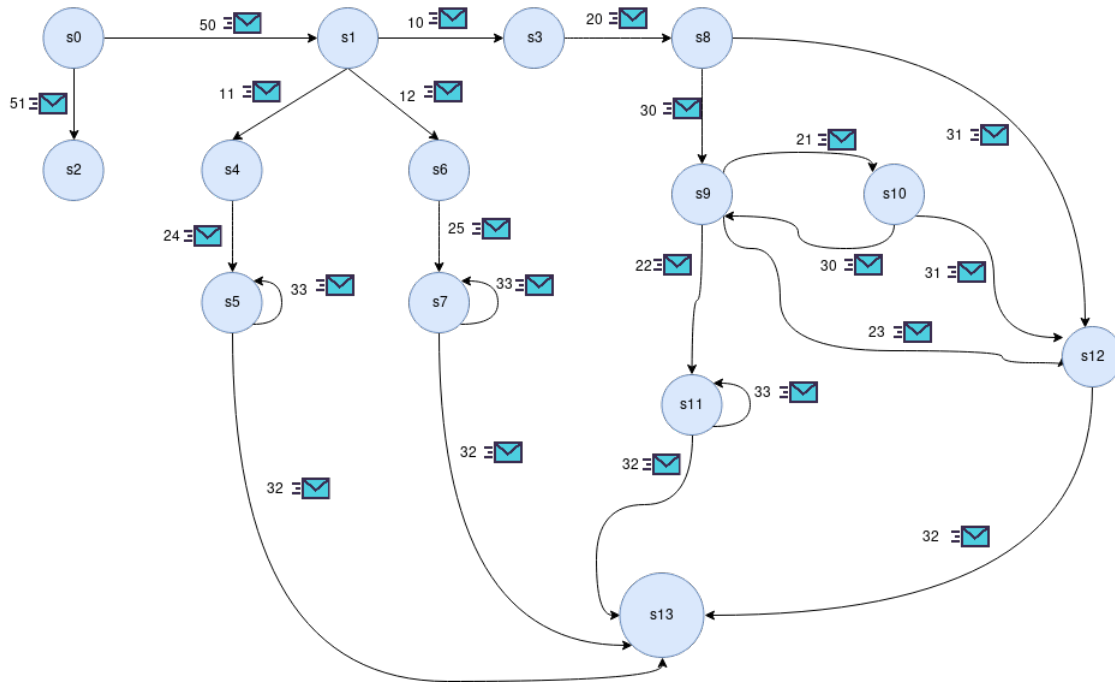
Servidor	
Código	Descripción
20	¿Capturar al Pokemon x?.
21	¿Intentar captura de nuevo? Quedan k intentos.
22	Envía Pokemon (imagen) capturado.
23	Número de intentos de captura agotados.

Cliente	
Código	Descripción
10	Solicitar al servidor por parte del cliente, un Pokemon para capturar.

Y los estados representaban lo siguiente:

Estados	
Código	Descripción
s0	Estado inicial, desde aquí comienza la conexión del protocolo de la capa de aplicación.
s1	Recibe solicitud del cliente, ofrece aleatoriamente un Pokemon para capturar.
s2	Indica si quiere capturar o no el Pokemon ofrecido.
s3	Inicia un contador con {n} como el máximo número de intentos. Aleatoriamente indica si se capturó al Pokemon o no.
s4	Da respuesta para reintentar captura de Pokemon.
s5	Se recibe Pokemon capturado (imagen).
s6	Terminando la sesión.
s7	Cierre de conexión.

Tomado en cuenta esta base, diseñamos el siguiente protocolo. Veamos primero el autómata de este nuevo protocolo:



Cuya tabla de transiciones es:

	10	11	12	20	21	22	23	24	25	30	31	32	33	40	41	50	51
s0																s1	s2
s1	s3	s4	s6														
s2																	
s3				s8													
s4								s5									
s5												s13	s5				
s6									s7								
s7												s13	s7				
s8										s9	s12						
s9					s10	s11	s12										
s10										s9	s12						
s11												s13	s11				
s12												s13					
s13																	

Para el cual agregamos algunos mensajes y mantuvimos los que ya existían, así quedando definidos de la siguiente manera:

En común	
Código	Descripción
30	Sí.
31	No.
32	Terminando sesión.
33	Recibido (Ack).

Servidor	
Código	Descripción
20	¿Capturar al Pokemon x?.
21	¿Intentar captura de nuevo? Quedan k intentos.
22	Envía Pokemon (imagen) capturado.
23	Número de intentos de captura agotados.
24	Envía Pokedex.
25	Envía catálogo.

Cliente	
Código	Descripción
10	Solicitar al servidor por parte del cliente, un Pokemon para capturar.
11	Solicitar al servidor por parte del cliente, ver su Pokedex.
12	Solicitar al servidor por parte del cliente, ver el catálogo.

Además agregamos dos nuevos grupos de códigos: 40 para errores y 50 para funcionalidades del login.

Errores	
Código	Descripción
50	Acceso permitido.
51	Acceso denegado.

Login	
Código	Descripción
40	Se rechaza la solicitud de conexión
41	Se cierra la conexión por timeout. Espera finalizada.

Como se modifico un poco el diseño de los mensajes y el flujo de comunicación entre el servidor y el cliente, los estados quedaron definidos de la siguiente manera:

Estados	
Código	Descripción
s0	Estado inicial, desde aquí comienza la conexión del protocolo de la capa de aplicación. Se solicitan credenciales para ingresar.
s1	Acceso concedido. Entrada a la aplicación. Se indica si se desea atrapar un Pokemon o revisar el catálogo y el Pokedex
s2	Acceso denegado.
s3	Se recibe una solicitud para capturar un Pokemon, y manda el id del Pokemon a capturar.
s4	Se recibe una solicitud para ver el Pokedex.
s5	El servidor comienza a transferir el Pokedex al cliente, muestra el Pokedex al usuario
s6	Se recibe una solicitud para ver el catálogo de Pokemones.
s7	El servidor comienza a transferir el catalogo al cliente, muestra el catálogo de Pokemones al usuario.
s8	Indica si quiere capturar o no el Pokemon ofrecido
s9	Se inicia un contador con el máximo número de intentos, e indica aleatoriamente si se capturó el Pokemon o no
s10	Da respuesta para reintentar captura de Pokemon
s11	El cliente recibe la imagen del servidor, se muestra imagen del Pokemon capturado
s12	Terminando la sesión.
s13	Cierre de conexión.

Finalmente, los mensajes se conforman de la siguiente manera.

Todos los mensajes para los códigos 10,11,12,23,30,31,32,33,40,41,50,51 estarán conformados por un byte, es decir:

code
1 byte

El mensaje 20, de la siguiente manera:

code	idPokemon
1 byte	1 byte

El mensaje 21, de la siguiente manera:

code	idPokemon	numAttempts
1 byte	1 byte	1 bytes

El mensaje 22, de la siguiente manera:

code	idPokemon	imageSize	image
1 byte	1 byte	4 bytes	k bytes

El mensaje 24, de la siguiente manera:

code	pokedexSize	Pokedex
1 byte	4 bytes	k bytes

El mensaje 25, de la siguiente manera:

code	catálogoSize	Catálogo
1 byte	4 bytes	k bytes

¿Cómo usar?

Requisitos de preinstalación

Se asumirá que se utiliza un SO Debian para la instalación.

Se requieren los siguientes programas para la correcta instalación y funcionamiento de la aplicación:

- Python versión 3.

- Manejador de paquetes *pip3*
- Manejador de bases de datos *MySQL*

Además, sin importar si se fungirá la función de cliente o servidor, el usuario del equipo que se ejecutará debe estar dentro del grupo *sudoers*. Esto debido a que la instalación de la documentación en Linux (man) se realiza en la ruta */usr/share/man/man7*

¿Quieres instalar?

- **Servidor:** Dirigirse a la carpeta `./Instalaciones` y ejecutar `./make_servidor -i`
Es importante que al momento que se esté haciendo la instalación para la base de datos se ocupen las credenciales: **doggos - doggos2020**, usuario y contraseña, respectivamente.
- **Cliente:** Dirigirse a la carpeta `./Instalaciones` y ejecutar `./make_cliente -i`

¿No es la primera vez?

Es importante señalar que primero inicializamos el servidor, y después los clientes pueden iniciar una conexión. Entonces, para cada uno de éstos se debe hacer lo siguiente:

- **Para el servidor**
 - En una terminal, nos situamos en la ubicación del archivo `pokemonServer.py`
 - No necesitamos parámetros extra para ejecutar el servidor. `./pokemonServer.py`
- **Para el cliente**
 - En una terminal, nos situamos en la ubicación del archivo `pokemonClient.py`
 - Este programa recibe como parámetros iniciales la dirección IP a través de la cual se quiere conectar, y el puerto. Por lo tanto, ejecutamos de la siguiente manera: `./pokemonClient.py <IP> <port>`

¿Quieres desinstalar?

- **Servidor:** Dirigirse a la carpeta `./Instalaciones` y ejecutar `./make_servidor -d`
- **Cliente:** Dirigirse a la carpeta `./Instalaciones` y ejecutar `./make_cliente -d`

A continuación mostraremos, a forma de tutorial, como ejecutar tanto el servidor como el cliente. Además mostraremos como, por parte del cliente, interactuar con la aplicación.

Paso a Paso:

Como ya vimos, primero debemos de inicializar el servidor; para lo cual hacemos lo siguiente:

```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonServer.py
Socket created
Socket now listening
```

A continuación podemos ejecutar el cliente  e ingresar nuestras credenciales para comenzar el juego.

¹En realidad se pueden ejecutar varios clientes a la vez. El servidor puede atender hasta diez simultáneamente

```

valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
>> Paulo
Ingrese la contraseña
>>

  _ _ _ _ _
 | _ _ \ / _ \ | / _ \ _ _ \ / _ \ | | | | | | |
 | | _ | ( ) | < _ _ | | | ( ) | |
 | _ _ \ _ _ / | \ _ _ | | | \ _ _ / |
 | _ |
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
>> █

```

Posteriormente podemos escoger cualquiera de las tres opciones para realizar alguna acción respecto al juego, o podemos elegir salir de la aplicación. Mostraremos que hacer en cada una de las opciones.

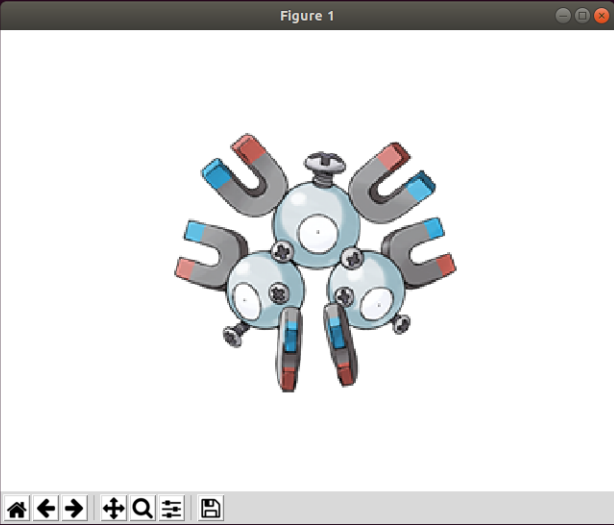
Veamos, primero, que hacer en caso de que deseemos atrapar un Pokemon. Para esto debemos, necesariamente, ingresar la opción *P*. Al hacer esto la aplicación nos ofrecerá un Pokemon para capturar, nosotros podemos escoger si capturarlo o no. En caso de que lo capturemos nos mandará la imagen del Pokemon capturado; es importante aclarar que antes de seguir con el flujo del programa debemos cerrar el visor de la imagen, podemos guardarla si así lo deseamos. En otro caso podemos volver a intentar capturarlo (siempre y cuando tengamos intentos disponibles).

```

valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
>> Paulo
Ingrese la contraseña
>>

  _ _ _ _ _
 | _ _ \ / _ \ | / _ \ _ _ \ / _ \ | | | | | | |
 | | _ | ( ) | < _ _ | | | ( ) | |
 | _ _ \ _ _ / | \ _ _ | | | \ _ _ / |
 | _ |
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
>> P
¿Capturar al Pokemon Magnetron?
Sí [S] o No [N]
>> S
¿Intentar captura de nuevo? Quedan 4 intentos
Sí [S] o No [N]
>> S
¿Intentar captura de nuevo? Quedan 3 intentos
Sí [S] o No [N]
>> S
¿Intentar captura de nuevo? Quedan 2 intentos
Sí [S] o No [N]
>> S
Capturaste al pokemon...
█

```



Si no lo capturamos, resultaría:

```

valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
>> Alma
Ingrese la contraseña
>>

  _
 | _ _ \ / _ _ \ | / / _ _ \ / _ _ \
 | |_) | ( ) | < _ _ \ | | | | ( ) | | | |
 | _ _ \ / _ _ \ | / / _ _ \ | | | | / _ _ \ | | | |
 | _ |
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
>> P
¿Capturar al Pokemon Poliwhag?
Sí [S] o No [N]
>> S
¿Intentar captura de nuevo? Quedan 1 intentos
Sí [S] o No [N]
>> S
Te quedaste sin intentos :(
Terminando conexión...

```

Luego, podemos revisar el Pokedex² de la siguiente manera:

```

valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonClient.py 127.0.0.1 9999
Ingrese el nombre de usuario con el que está registrado
>> Paulo
Ingrese la contraseña
>>

  _
 | _ _ \ / _ _ \ | / / _ _ \ / _ _ \
 | |_) | ( ) | < _ _ \ | | | | ( ) | | | |
 | _ _ \ / _ _ \ | / / _ _ \ | | | | / _ _ \ | | | |
 | _ |
Bienvenido a Pokemon Go! ¿Deseas capturar un Pokémon [P], revisar el Pokedex [X], revisar el catálogo? [C] o salir [S]?
>> X
Mostrando Pokedex...
Bellsprout, Koffing,
Farfetchd, Drowzee,

```

Finalmente, podemos revisar el catálogo³ de todos los Pokemones disponibles en la aplicación.

²Es decir, podemos consultar los Pokemones que hemos capturado

³Esta funcionalidad puede tardar unos momentos

[illegible]

Por el lado del servidor veremos que aunque se muera la conexión éste seguirá escuchando y atendiendo peticiones. Por lo cual sólo veríamos algo como:

```
valde@ubuntu:~/Documents/7to semestre/Redes/proyectoTCP/Proyecto2-Pokemon$ ./pokemonServer.py 127.0.0.1
Socket created
Socket now listening
Connected with 127.0.0.1:56332
Tiempo de respuesta excedido: 10 segundos
```

Observación de su funcionamiento mediante capturas de Wireshark

Profundizando en el funcionamiento interno del protocolo, se mostrarán lecturas capturadas con el sniffer *Wireshark* en el proceso de las distintas funciones de la aplicación.

Veamos el envío de mensajes al momento de ingresar a la aplicación, donde vemos interacción con la base de datos en MySQL. Notemos que en el área de datos se visualizan datos de la conexión y el query mandado:

[illegible]

No.	Time	Source	Destination	Protocol	Length	Info
10	0.002020362	:::1	:::1	TCP	86	44712 → 3306 [ACK] Seq=1 Ack=94 Win=43776 Len=0 TSval=1016101828 TSecr=1016101828
11	0.003531550	:::1	:::1	MySQL	394	Login Request user=doggos db=TCP201-Pokemon
12	0.003549850	:::1	:::1	TCP	86	3306 → 44712 [ACK] Seq=94 Ack=309 Win=44800 Len=0 TSval=1016101828 TSecr=1016101828
13	0.003735025	:::1	:::1	MySQL	116	Response OK
14	0.004082250	:::1	:::1	MySQL	139	Request Query
15	0.004392000	:::1	:::1	MySQL	197	Response OK
16	0.004602025	:::1	:::1	MySQL	108	Request Query
17	0.004702975	:::1	:::1	MySQL	197	Response OK
18	0.004802587	:::1	:::1	MySQL	107	Request Query
19	0.004930887	:::1	:::1	MySQL	116	Response OK
20	0.005049138	:::1	:::1	MySQL	91	Request Ping
21	0.005117275	:::1	:::1	MySQL	97	Response OK
22	0.005492238	:::1	:::1	MySQL	143	Request Query
23	0.006353813	:::1	:::1	MySQL	248	Response
24	0.007072475	127.0.0.1	127.0.0.1	TCP	67	9999 → 53422 [PSH, ACK] Seq=2 Ack=15 Win=342 Len=1 TSval=523515613 TSecr=523515606
25	0.007198475	:::1	:::1	MySQL	91	Request Quit
26	0.007258138	:::1	:::1	TCP	86	44712 → 3306 [FIN, ACK] Seq=472 Ack=549 Win=44800 Len=0 TSval=1016101832 TSecr=1016101831
27	0.007882525	:::1	:::1	TCP	86	3306 → 44712 [FIN, ACK] Seq=549 Ack=473 Win=44800 Len=0 TSval=1016101833 TSecr=1016101832
28	0.007926763	:::1	:::1	TCP	86	44712 → 3306 [ACK] Seq=473 Ack=550 Win=44800 Len=0 TSval=1016101833 TSecr=1016101833
29	0.048541092	127.0.0.1	127.0.0.1	TCP	66	53422 → 9999 [ACK] Seq=15 Ack=3 Win=342 Len=0 TSval=523515654 TSecr=523515613
MySQL Protocol						
Packet Length: 16						
Packet Number: 4						
Catalog: Alma						
Database: doggos2020						
[Malformed Packet: MySQL]						
0030	00 00 00 00 00 01 0c ea	ae a8 8f 08 c0 15 3a be:.			
0040	ed 4e 80 18 01 5e 00 ca	00 00 01 01 08 0a 3c 90	..N...A...<.			
0050	7b c7 3c 90 7b c6 01 00	00 01 02 3e 00 00 02 03	{<{<...>...			
0060	64 65 66 0f 54 43 50 32	30 31 2d 50 6f 0b 65 6d	def-TCP2 01-Pokem			
0070	6f 6e 07 55 73 75 61 72	69 6f 07 55 73 75 61 72	on-Usuar 1o-Usuar			
0080	69 6f 06 4e 6f 6d 62 72	65 06 4e 6f 6d 62 72 65	io-Nombr e-Nombr			
0090	0c 2d 00 b4 00 00 00 fd	05 50 00 00 00 38 00 00P...8..			
00a0	03 03 64 65 66 0e 54 43	50 32 30 31 2d 50 6f 6b	..def-TC P201-Pok			
00b0	65 6d 6f 6e 07 55 73 75	61 72 69 6f 07 55 73 75	emon-Usu ario-Usu			
00c0	61 72 69 6f 03 50 77 64	03 50 77 64 0c 2d 00 40	ario-Pwd ..@			
00d0	00 00 00 fd 01 10 00 00	00 10 00 00 04 04 41 6cAl			
00e0	6d 61 0a 05 0f 07 05 6f	73 32 30 32 50 07 00 00	ma-doggo s2020...			
00f0	05 fe 00 00 01 00 00 00				

Mensajes enviados durante la captura de un Pokemon

No.	Time	Source	Destination	Protocol	Length	Info
53	3.608240474	:::1	:::1	TCP	86	3306 → 53422 [FIN, ACK] Seq=483 Ack=467 Win=44800 Len=0 TSval=1013984793 TSecr=1013984793
54	3.608265386	:::1	:::1	TCP	86	44600 → 3306 [ACK] Seq=467 Ack=484 Win=43776 Len=0 TSval=1013984793 TSecr=1013984793
55	3.608331349	127.0.0.1	127.0.0.1	TCP	75	9999 → 53308 [PSH, ACK] Seq=5 Ack=16 Win=342 Len=9 TSval=521398574 TSecr=521398566
56	3.608361761	127.0.0.1	127.0.0.1	TCP	66	53308 → 9999 [ACK] Seq=16 Ack=14 Win=342 Len=0 TSval=521398574 TSecr=521398574
57	5.634861639	127.0.0.1	127.0.0.1	TCP	67	53308 → 9999 [PSH, ACK] Seq=16 Ack=14 Win=342 Len=1 TSval=521400600 TSecr=521398574
58	5.635155576	127.0.0.1	127.0.0.1	TCP	68	9999 → 53308 [PSH, ACK] Seq=14 Ack=17 Win=342 Len=2 TSval=521400601 TSecr=521400600
59	5.635209939	127.0.0.1	127.0.0.1	TCP	66	53308 → 9999 [ACK] Seq=17 Ack=16 Win=342 Len=0 TSval=521400601 TSecr=521400601
60	5.635993789	127.0.0.1	127.0.0.1	TCP	67	53308 → 9999 [PSH, ACK] Seq=17 Ack=16 Win=342 Len=1 TSval=521400601 TSecr=521400601
61	5.636160564	127.0.0.1	127.0.0.1	TCP	70	9999 → 53308 [PSH, ACK] Seq=16 Ack=18 Win=342 Len=4 TSval=521400602 TSecr=521400602
62	5.636332601	127.0.0.1	127.0.0.1	TCP	67	53308 → 9999 [PSH, ACK] Seq=18 Ack=20 Win=342 Len=1 TSval=521400602 TSecr=521400602
63	5.636574776	127.0.0.1	127.0.0.1	TCP	21954	9999 → 53308 [ACK] Seq=20 Ack=19 Win=342 Len=21888 TSval=521400602 TSecr=521400602
64	5.636681464	127.0.0.1	127.0.0.1	TCP	21954	9999 → 53308 [PSH, ACK] Seq=21908 Ack=19 Win=342 Len=21888 TSval=521400602 TSecr=521400602
Frame 62: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface 0						
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)						
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1						
Transmission Control Protocol, Src Port: 53308, Dst Port: 9999, Seq: 18, Ack: 20, Len: 1						
Data (1 byte)						
Data: 21						
[Length: 1]						
0000	00 00 00 00 00 00 00 00	00 00 00 00 00 00 45 00E.....			
0010	00 35 68 7b 40 00 00 06	d4 45 7f 00 00 01 7f 00	..5h(@ @ ..E.....			
0020	00 01 d0 3c 27 0f 65 49	62 48 1e fa 4c 1a 80 18	...<' eI bH ..L...			
0030	01 56 fe 29 00 00 01 01	00 0a 1f 13 f1 1a 1f 13	..V.).....			
0040	f1 1a 21		..!			

Mensajes enviados durante la consulta del Pokedex

Capturing from Loopback: lo

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
26	5.043723104	:::1	:::1	MySQL	248	Response
27	5.044061429	127.0.0.1	127.0.0.1	TCP	67	9999 → 53372 [PSH, ACK] Seq=2 Ack=15 Win=43776 Len=1 TSval=522502743 TSecr=522502729
28	5.044114554	:::1	:::1	MySQL	91	Request [QUIT]
29	5.044137579	:::1	:::1	TCP	86	44662 → 3306 [FIN, ACK] Seq=472 Ack=549 Win=44800 Len=0 TSval=1015088962 TSecr=1015088961
30	5.044313254	:::1	:::1	TCP	86	3306 → 44662 [FIN, ACK] Seq=549 Ack=473 Win=44800 Len=0 TSval=1015088962 TSecr=1015088962
31	5.044329079	:::1	:::1	TCP	86	44662 → 3306 [ACK] Seq=473 Ack=550 Win=44800 Len=0 TSval=1015088962 TSecr=1015088962
32	5.085878570	127.0.0.1	127.0.0.1	TCP	66	53372 → 9999 [ACK] Seq=15 Ack=3 Win=43776 Len=0 TSval=522502784 TSecr=522502743
33	7.868835411	127.0.0.1	127.0.0.1	TCP	67	53372 → 9999 [PSH, ACK] Seq=15 Ack=3 Win=43776 Len=1 TSval=522505567 TSecr=522502743
34	7.870186574	:::1	:::1	TCP	94	44666 → 3306 [SYN] Seq=0 Win=43690 Len=0 MSS=65476 SACK_PERM=1 TSval=1015091788 TSecr=0 WS=128
35	7.870230799	:::1	:::1	TCP	94	3306 → 44666 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65476 SACK_PERM=1 TSval=1015091788 TSecr=1015091788
36	7.870270536	:::1	:::1	TCP	86	44666 → 3306 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=1015091788 TSecr=1015091788
37	7.870605724	:::1	:::1	MySQL	179	Server Greeting proto=10 version=5.5.5-10.4.10-MariaDB

Frame 27: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 9999, Dst Port: 53372, Seq: 2, Ack: 15, Len: 1
Data (1 byte)
Data: 32
[Length: 1]

0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00E
0010 00 35 0d 86 40 00 00 06 2f 3b 7f 00 00 01 7f 00 ..5..0..0..;/.....
0020 00 01 27 0f d0 7c 22 97 29 35 fa 96 21 f2 80 18|".)5.....
0030 01 56 fe 29 00 00 01 01 08 0a 1f 24 c2 57 1f 24 ..V.).....\$W\$
0040 c2 49 3212

Loopback: lo: <live capture in progress> Packets: 70 · Displayed: 70 (100.0%) Profile: Default

Mensajes enviados durante la consulta del catálogo de Pokemones

Capturing from Loopback: lo

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
46	8.656690016	:::1	:::1	MySQL	107	Request Query
47	8.656859091	:::1	:::1	MySQL	116	Response OK
48	8.656964679	:::1	:::1	MySQL	91	Request Ping
49	8.657060704	:::1	:::1	MySQL	97	Response OK
50	8.657278016	:::1	:::1	MySQL	117	Request Query
51	8.658129654	:::1	:::1	MySQL	2033	Response
52	8.658189154	:::1	:::1	TCP	86	44678 → 3306 [ACK] Seq=441 Ack=2334 Win=174720 Len=0 TSval=1015237823 TSecr=1015237823
53	8.659474767	127.0.0.1	127.0.0.1	TCP	67	9999 → 53386 [PSH, ACK] Seq=3 Ack=16 Win=43776 Len=1 TSval=522651605 TSecr=522651596
54	8.659507729	127.0.0.1	127.0.0.1	TCP	66	53386 → 9999 [ACK] Seq=16 Ack=4 Win=43776 Len=0 TSval=522651606 TSecr=522651605
55	8.659641642	127.0.0.1	127.0.0.1	TCP	67	53386 → 9999 [PSH, ACK] Seq=16 Ack=4 Win=43776 Len=1 TSval=522651606 TSecr=522651605
56	8.659715804	127.0.0.1	127.0.0.1	TCP	70	9999 → 53386 [PSH, ACK] Seq=4 Ack=17 Win=43776 Len=4 TSval=522651606 TSecr=522651606
57	8.659774717	127.0.0.1	127.0.0.1	TCP	67	53386 → 9999 [PSH, ACK] Seq=17 Ack=8 Win=43776 Len=1 TSval=522651606 TSecr=522651606
58	8.659845754	127.0.0.1	127.0.0.1	TCP	67	9999 → 53386 [PSH, ACK] Seq=8 Ack=18 Win=43776 Len=1 TSval=522651606 TSecr=522651606
59	8.659984817	127.0.0.1	127.0.0.1	TCP	67	53386 → 9999 [PSH, ACK] Seq=18 Ack=9 Win=43776 Len=1 TSval=522651606 TSecr=522651606
60	8.660043042	127.0.0.1	127.0.0.1	TCP	75	9999 → 53386 [PSH, ACK] Seq=9 Ack=19 Win=43776 Len=9 TSval=522651606 TSecr=522651606
61	8.701010508	127.0.0.1	127.0.0.1	TCP	66	53386 → 9999 [ACK] Seq=19 Ack=18 Win=43776 Len=0 TSval=522651648 TSecr=522651606
62	8.701922208	127.0.0.1	127.0.0.1	TCP	67	9999 → 53386 [PSH, ACK] Seq=18 Ack=19 Win=43776 Len=1 TSval=522651648 TSecr=522651648
63	8.701955208	127.0.0.1	127.0.0.1	TCP	66	53386 → 9999 [ACK] Seq=19 Ack=19 Win=43776 Len=0 TSval=522651648 TSecr=522651648
64	8.702141508	127.0.0.1	127.0.0.1	TCP	67	53386 → 9999 [PSH, ACK] Seq=19 Ack=19 Win=43776 Len=1 TSval=522651648 TSecr=522651648
65	8.702335208	127.0.0.1	127.0.0.1	TCP	73	9999 → 53386 [PSH, ACK] Seq=19 Ack=20 Win=43776 Len=7 TSval=522651648 TSecr=522651648

Frame 60: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 9999, Dst Port: 53386, Seq: 9, Ack: 19, Len: 9
Data (9 bytes)
Data: 42756c626173617572

0000 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00E
0010 00 3d e3 3d 40 00 00 06 59 7b 7f 00 00 01 7f 00 ..=..0..0..Y{.....
0020 00 01 27 0f d0 4a 43 ad 2d 09 1f 1c 95 20 80 18C.....
0030 01 56 fe 31 00 00 01 01 08 0a 1f 27 07 d6 1f 27 ..V.1.....'.....
0040 07 d6 42 75 6c 62 61 73 61 75 72 ..Bulbasaur

Loopback: lo: <live capture in progress> Packets: 812 · Displayed: 812 (100.0%) Profile: Default

Usuarios:

Los usuarios dados de alta para la ejecución de la aplicación son los siguientes:

Usuario	Contraseña
Alma	doggos2020
Paulo	profesor
Ismael	ayudante.lab
Ulises	ayudante.clase
Alejandro	doggos2020

Funciones usadas en la programación:

Para revisar la lista de las funciones, ver el apartado de la documentación de las mismas al final del documento.

2.1 Cliente Pokemon Go!

Implementación de un cliente para el juego Pokemon Go! e interactúa directamente con el usuario

`pokemonClient.cerrarSesion(soc)`

Cierre normal de sesión del usuario.

Parámetros `soc` (*Socket*) – Socket de la conexión

Devuelve Nada

`pokemonClient.displayCatalogo(catalogo)`

Imprime en pantalla el catálogo de Pokemones disponibles de manera «amigable».

Parámetros `catalogo` (*List of String*) – Catalogo de Pokemones

Devuelve Nada

`pokemonClient.displayPokedex(pokedex)`

Imprime en pantalla el Pokedex de manera «amigable».

Parámetros `pokedex` (*List of String*) – Pokedex de Pokemones

Devuelve Nada

`pokemonClient.login(soc)`

Transfiere los datos al servidor para validar el acceso, y cierra el programa si los datos no son válidos.

Parámetros `soc` (*Socket*) – Socket de la conexión

Devuelve Nada

`pokemonClient.main()`

Función principal

`pokemonClient.muestraCatalogo (soc)`

Le muestra el catálogo disponible de Pokemones al usuario.

Parámetros `soc` (*Socket*) – Socket de la conexión

Devuelve Nada

`pokemonClient.muestraPokedex (soc)`

Muestra el Pokedex del usuario que solicita esta acción al usuario.

Parámetros `soc` (*Socket*) – Socket de la conexión

Devuelve Nada

`pokemonClient.muestraPokemon (bytes)`

Despliega el pokemon asignado.

Parámetros `bytes` (*bytearray*) – bytes de la imagen del pokemon a desplegar

Devuelve Nada

`pokemonClient.playPokemon (soc)`

Permite que el usuario juegue Pokemon Go.

Parámetros `soc` (*Socket*) – Socket de la conexión

Devuelve Nada

`pokemonClient.printPokemon ()`

Imprime en pantalla el logo Pokemon :returns: Nada

`pokemonClient.terminarConTimeout (soc)`

Termina la conexión pues el tiempo de espera de la respuesta del Servidor ha excedido.

Parámetros `soc` (*Socket*) – Socket de la conexión

Devuelve Nada

`pokemonClient.terminarConexion ()`

Termina la conexion pues el Servidor notifica que el tiempo de espera ha excedido.

Param Nada

Devuelve Nada

2.2 Servidor Pokemon Go!

Implementación de un servidor para el juego Pokemon Go!

`pokemonServer.avisTimeout (connection)`

Manda el mensaje de cierre de sesión al cliente por tiempo de espera excedido.

Parámetros `connection` (*Conexión*) – Conexión entre el cliente y el servidor

Devuelve Nada

`pokemonServer.cerrarSesion (connection)`

Cierre de sesión entre el servidor y el cliente al cual le pertenece la conexión.

Parámetros `connection` (*Conexión*) – Conexión entre el cliente y el servidor

Devuelve Nada

`pokemonServer.clientThread(connection, ip, port, max_buffer_size=5120)`

Manejador del hilo que sostiene la conexión entre el servidor y un cliente

Parámetros

- **connection** (*Conexión*) – Conexión entre el servidor y el cliente que abrió el hilo
- **ip** (*String*) – Dirección IP de la conexión
- **port** (*Integer*) – Puerto a través del cual el servidor mantiene la conexión con el cliente
- **max_buffer_size** (*Integer*) – Número máximo de bytes que puede recibir en un paquete del cliente

Devuelve Nada

`pokemonServer.getNombrePokemon(idPokemon)`

Regresa el nombre del pokemon dado su id.

Parámetros `idPokemon` (*Integer*) – Id del Pokemon a capturado

Devuelve String

`pokemonServer.giveAccess(connection, max_buffer_size=5120)`

Autentifica a usuarios registrados y proporciona acceso a la ejecución de la aplicación

Parámetros

- **connection** (*Conexión*) – Conexión entre el servidor y el cliente que abrió el hilo
- **max_buffer_size** (*Integer*) – Número máximo de bytes que puede recibir en un paquete del cliente

Devuelve Integer, String -> Valor que representa la correctud del acceso; Cadena que representa al usuario activo.

`pokemonServer.guardaEnPokedex(idPokemon, user)`

Guarda el pokemon capturado en el pokedex del usuario.

Parámetros

- **idPokemon** (*Integer*) – Id del Pokemon a capturado
- **user** (*String*) – Usuario que capturo

Devuelve Nada

`pokemonServer.main()`

Función principal.

`pokemonServer.muestraCatalogo(connection)`

Muestra el catalogo.

Parámetros `connection` (*Conexión*) – Conexión entre el servidor y el cliente

Devuelve Nada

`pokemonServer.muestraPokedex(connection, user)`

Muestra el Pokedex del usuario.

Parámetros

- **connection** (*Conexión*) – Conexión entre el servidor y el cliente
- **user** (*String*) – Usuario que capturo

Devuelve Nada

`pokemonServer.playPokemonGo(connection, user)`

Método que simula el comportamiento del juego Pokemon Go.

Parámetros **connection** (*Conexión*) – Conexión entre el servidor y el cliente

Devuelve Nada

`pokemonServer.start_server()`

Inicialización del servidor

Parámetros **ip_dir** (*String*) – Dirección IP del socket al cual se va conectar el servidor

Devuelve Nada

`pokemonServer.terminarConexion()`

Termina la conexion pues el Cliente notifica que el tiempo de espera ha excedido.

Param Nada

Devuelve Nada